

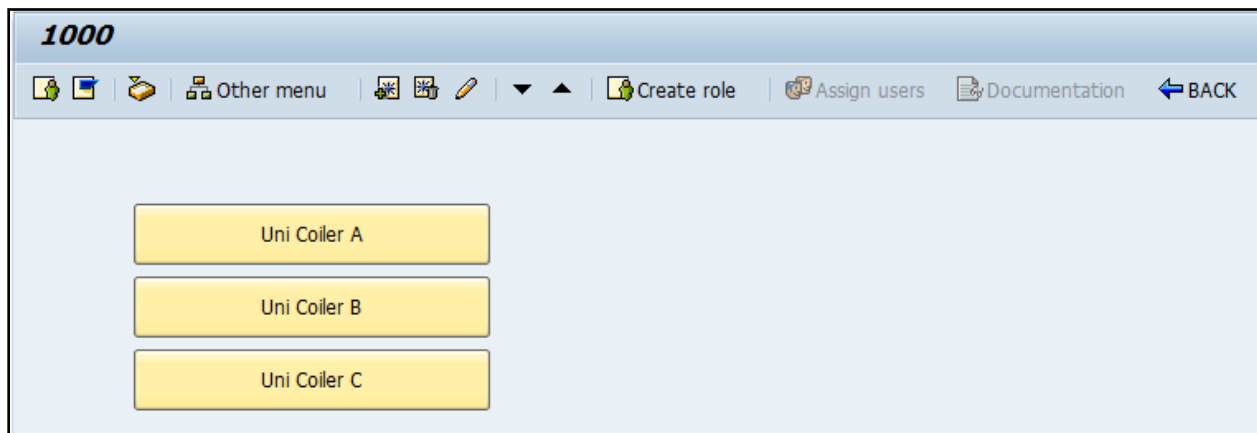
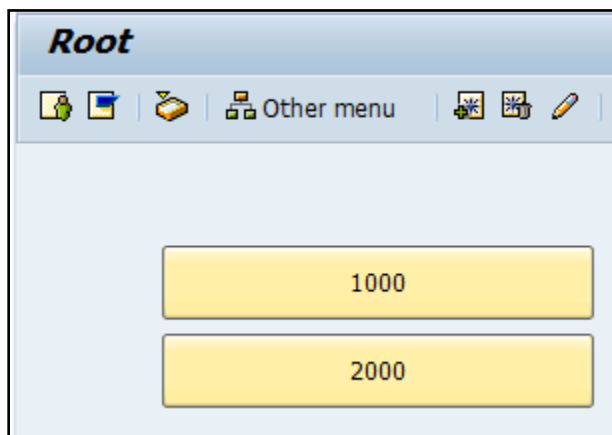
Liquid UI: Tree based multilevel screens

Purpose



Create UI dynamically on the same screen based on data passed as an array by converting to hierarchical tree structure.


User Interface

Log into SAP and on the SAP Easy Access Screen click on the first pushbutton and navigate all the way to land on MM02 screen.








Uni Coiler A





Other menu





Create role

Assign users



Documentation


BACK

Uni Coiler A South East




Uni Coiler A North East



Uni Coiler A South East





Other menu





Create role



Assign users


Documentation

BACK




Uni Coiler A South East 1



Uni Coiler A South East 1





Other menu





Create role

Assign users

Documentation

BACK

Uni Coiler A South East 1A


Uni Coiler A South East 1B

Change Material (Initial Screen)

Select view(s) Organizational levels Data

Material

K1-E-SE-1A



Change Number

Liquid UI Code [Script]

SAPLSMTR NAVIGATION.E0100.sjs -

```
SAPLSMTR_NAVIGATION.E0100.sjs
1  load('commonFunctions.sjs');    // Loads the functions file
2
3  // This data may come from -
4  // a. Fixed File
5  // b. RFC Call
6  // The data from File or RFC needs to be built into an array similar to below
7  // "Parent,Child,Value,Tcode"
8  testData = [];
9  testData[0] = "Root,1000,K1,";
10 testData[1] = "1000,Uni Coiler A,K1-A,";
11 testData[2] = "1000,Uni Coiler B,K1-B,";
12 testData[3] = "1000,Uni Coiler C,K1-C,";
13 testData[4] = "Uni Coiler A,Uni Coiler A South East,K1-E-SE,";
14 testData[5] = "Uni Coiler A,Uni Coiler A North East,K1-E-NE,";
15 testData[6] = "Uni Coiler A South East,Uni Coiler A South East 1,K1-E-SE-1,";
16 testData[7] = "Uni Coiler A South East 1,Uni Coiler A South East 1A,K1-E-SE-1A,MM02";
17 testData[8] = "Uni Coiler A South East 1,Uni Coiler A South East 1B,10030388,IW22";
18 testData[9] = "Root,1000,100,";
19 testData[10] = "Root,"; // ROOT - Not displayed on screen
20 testData[11] = "2000,TEST_CASING,100-100,MM03";
21
22
23 // User Interface
24 clearscreen();
25 if(isBlank(first_time)){
26     first_time = "X";
27     createTree(testData,""); // Pass separator to the function and use that (Eg: '^','-','^^')
28 }
29 drawUI();
30
```

commonFunctions.sjs -

```
commonFunctions.sjs
1  // Function to trim blank spaces at the end of the string
2  String.prototype.trim=function(){return this.replace(/^\s+|\s+$/g,'');}
3
4  // Function to check if the string value is blank
5  function isBlank(jvar){
6      if(typeof jvar == 'string') {
7          jvar = jvar.trim();
8      }
9      if(typeof jvar == 'undefined') {
10         jvar = '';
11     }
12     return(jvar == 'undefined' || jvar == undefined || jvar == null || jvar == "" || jvar == void 0);
13 }
14
15 // Function to paint the User Interface
16 function drawUI(){
17     if(!isBlank(current.getChildren())){
18         title(current.name);
19         for(x=0; x<current.getChildren().length; x++){
20             pushbutton([(x+1)*2,10], current.children[x].name, "?", {"size":[2,30], "process":changeCurrent, "using":{"new_current":current.children[x]}});
21         }
22
23         // If the option to go back is possible
24         if(current.getParentNode() != null){
25             pushbutton([TOOLBAR,"@9S@BACK","?", {"process":changeCurrent, "using":{"new_current":current.getParentNode()}});
26         }
27     } else{
28         tcode = current.txcode;
29         value = current.val;
30         if(!isBlank(tcode)){
31             enter({"process":navigateToTransaction, "using":{"t_tcode":tcode, "t_val":value}});
32         } else{
33             message('E: No transaction specified');
34             first_time = '';
35             enter('?');
36         }
37     }
38 }
39
40 // Function to perform some action on the last level
41 // Navigates to the transaction code and uses the value passed
42 function navigateToTransaction(param) {
43     var tcode = param.t_tcode;
44     first_time = '';
45     enter('/n'+tcode);
46
47     onscreen '='; // Rest of the navigation logic based on Transaction code
48     title(_title);
49     if(tcode == 'MM02'){
50         set('F[Material]',param.t_val);
51     }
52 }
```

```

51         } else if(tcode == 'MM03'){
52             set('F[Material]',param.l_val);
53             goto MM03_PROCESS;
54         } else if(tcode == 'IW22'){
55             set('F[Notification]',param.l_val);
56             goto IW22_PROCESS;
57         }
58         enter('?');
59         goto SCRIPT_END;
60
61         onscreen 'SAPLMGMM.0600'
62         MM03_PROCESS;;
63         enter();
64         goto SCRIPT_END;
65
66         onscreen 'SAPLIQS0.0100'
67         IW22_PROCESS;;
68         enter();
69
70         SCRIPT_END;;
71     }
72
73     // Function to initialize the Node containing name, value and transaction code
74     function Node(name,val,txcode){
75         this.name = name;
76         this.val = val;
77         this.txcode = txcode;
78         this.children = [];
79         this.parent = null;
80
81         this.setParentNode = function(node){
82             this.parent = node;
83         }
84         this.getParentNode = function(){
85             return this.parent;
86         }
87         this.addChild = function(node){
88             node.setParentNode(this);
89             this.children[this.children.length] = node;
90         }
91         this.getChildren = function(){
92             return this.children;
93         }
94     }
95
96     // Function to refresh and paint the new screen based on button click
97     function changeCurrent(param){
98         current = param.new_current;
99         enter("?");
100     }

```

```

102 // Function which creates the tree structure from an array and the data separator passed to it
103 function createTree(arr,separator){
104     // Create the root
105     for(i=0; i<arr.length; i++){
106         arrTemp = arr[i].split(separator);
107         if(isBlank(arrTemp[0])){
108             rootNode = new Node(arrTemp[1],arrTemp[2],arrTemp[3]);
109             arr.splice(i,1); // Delete the element in the array
110             break;
111         }
112     }
113
114     index = 0;
115     while(arr.length > 0){ // While elements are in the array
116         if(isBlank(arr[index])) // Reached the end of the data array, break out of while loop
117             break;
118
119         arrTemp = arr[index].split(separator);
120
121         result = inTree(rootNode,arrTemp[0]); // Check whether the Parent is in the Tree
122         if(!isBlank(result)){
123             result.addChild(new Node(arrTemp[1],arrTemp[2],arrTemp[3])); // Add the child
124             arr.splice(index,1); // Delete the element in the array
125             index = 0;
126         } else{
127             index++;
128         }
129     }
130     current = rootNode;
131 }
132
133 // Function which checks to see if the Parent Node exists in current tree
134 function inTree(rtnode,parent){
135     var nodeStack = [];
136     nodeStack.push(rtnode);
137
138     while(nodeStack.length > 0){ // Always 1 as 'nodeStack' contains the ROOT node
139         processNode = nodeStack.pop(); // 'processNode' points to the beginning of the node
140
141         if (processNode.name == parent) {
142             return processNode; // Returns once the node is found and adds to the tree
143             break;
144         } else if (processNode.children.length>0) {
145             for (ii = 0; ii < processNode.children.length; ii++) {
146                 nodeStack.push(processNode.children[ii]);
147             }
148         }
149     }
150     return ''; // Return an empty string if parent not found in the tree
151 }

```